Package 'SASmarkdown'

January 20, 2025

Version 0.8.2

Date 2023-03-09

Title 'SAS' Markdown

Description Settings and functions to extend the 'knitr' 'SAS' engine.

Imports knitr(>= 1.21), xfun(>= 0.4)

Suggests rmarkdown

SystemRequirements SAS

Maintainer Doug Hemken <dehemken@wisc.edu>

License MIT + file LICENSE

URL https://www.ssc.wisc.edu/~hemken/SASworkshops/sas.html#writing-sas-documentation

BugReports https://github.com/Hemken/SASmarkdown/issues

NeedsCompilation no

Author Doug Hemken [aut, cre] (SSCC, Univ. of Wisconsin-Madison),
Chao Cheng [ctb] (Statistician, Simcere Pharmaceutical Group Limited)

Repository CRAN

Date/Publication 2023-03-10 17:30:06 UTC

Contents

SASmarkdown-package *Settings and functions to extend the knitr SAS engine.*

Description

Using the "sas" language engine provided in knitr has a number of limitations. Each SAS code chunk is run as a separate batch file, and only the source code and the listing output are returned to the document being knit.

The functions in this package set up additional variations on the SAS language engine, enabling ODS HTML, HTML5, and LaTeX output to be returned to the document, as well as enabling SAS log output to be returned. These language engines are automatically created when the package is loaded.

When used with chunk option error=TRUE, the user can see some SAS errors automatically included in their document.

Another function here sets up a chunk hook, that repeats selected code chunks at the beginning of later code chunks. This allows the code in one chunk to use the results of a previous chunk. See sas_collectcode.

Another function sets up source hooks, allowing the user to suppress parts of the SAS log. See saslog_hookset.

The function spinsas processes SAS command files that include markup within SAS comments. See spinsas.

Author(s)

Doug Hemken

References

More documentation and examples: http://www.ssc.wisc.edu/~hemken/SASworkshops/sas.html#writing-sas-documentation

See Also

The package that this extends: knitr-package.

find_sas *A helper function that seeks to locate your SAS executable.*

Description

This function searches for recent versions of SAS (>= SAS 8), in some of the usual default installation locations.

This function is automatically invoked when the SASmarkdown library is attached - normally a user will not need to call this function.

In the event SAS is not found, you will have to specify the correct location yourself.

Usage

```
find_sas(message=TRUE)
```

Arguments

message	(logical) Whether or not to print a message when SAS is found.

Value

A character string with the path and name of the SAS executable.

Author(s)

Doug Hemken

See Also

SASmarkdown-package

Examples

```
indoc <- '
---
title: "Basic SASmarkdown Doc"
author: "Doug Hemken"
output: html_document
---
# An R console example
## In a first code chunk, set up with
```{r}
library(SASmarkdown)
```
Then mark SAS code chunks with
```{sas}
data class;
  set sashelp.class;
  bmi = 703*weight/height**2;
  run;

proc means;
  var bmi;
  run;
```
Some more document text here.
'

if (!is.null(SASmarkdown::find_sas())) {
  # To run this example, remove tempdir().
  fmd <- file.path(tempdir(), "test.md")
  fhtml <- file.path(tempdir(), "test.html")

  knitr::knit(text=indoc, output=fmd)
```

```
  rmarkdown::render(fmd, "html_document", fhtml)
}
```

`saslog_hookset` *A function to clean SAS log files*

Description

The main function here is `saslog_hookset`, which sets "hooks" for knitr. It can set a "source" hook to clean up SAS logs for the `saslog` engine, or set an "output" hook to clean up SAS logs written to files and read in using R code.

Used once per hook type per session (i.e. document), during set up.

Usage

```
saslog_hookset(hooktype)

sasloghook(x, options)
```

Arguments

`hooktype`	Declare which type of hook to set, "source" (the default) or "output".
`options`	`options` are passed to these functions when they are actually invoked within `knitr`.
`x`	The log text which is to be cleaned up

Details

The main function is used with either "source" or "output" as the value of `hooktype`.

The end user should not need to use `sasloghook` directly. This is a workhorse function used to process selected log output. The main use is when set up within `knit_hooks$set(source=sasloghook)`

Once this hook is set, the user may then set any chunk options

- SASproctime
- SASecho
- SASnotes

to FALSE to suppress that part of the SAS log.

Value

`saslog_hookset` is used for it's side effect of resetting a knitr hook.

`sasloghook` returns SAS log output internally to `knitr`.

Author(s)

Doug Hemken

See Also

knit_hooks

Examples

```
# saslog_hookset() # called during loading

indoc <- '
---
title: "Basic SASmarkdown Doc"
author: "Doug Hemken"
output: html_document
---
# In a first code chunk, set up with
```{r}
library(SASmarkdown)
```
Then set up SAS code chunks with
```{sas}
proc means data=sashelp.class;
run;
```
'
if (!is.null(SASmarkdown::find_sas())) {
  # To run this example, remove tempdir().
  fmd <- file.path(tempdir(), "test.md")
  fhtml <- file.path(tempdir(), "test.html")

  knitr::knit(text=indoc, output=fmd)
  rmarkdown::render(fmd, "html_document", fhtml)
}
```

---

sas_collectcode *Create a knitr chunk hook for accumlating code.*

Description

This wrapper function calls `knitr::knit_hooks$set()` to define a chunk hook. When the chunk hook is later invoked, this writes the contents of the current chunk to the end of a SAS `autoexec.sas` file.

This may be used with any of the SAS language engines.

Usage

```
sas_collectcode()
```

Details

This function is automatically invoked when the SASmarkdown library is attached. Normally a user will not need to call this function, instead using `collectcode=TRUE` as a chunk option - see the example below.

When knitr calls SAS, each code chunk is processed as a separate SAS batch job. Where code in one chunk depends upon the results from a prevous chunk, code needs to be repeated and re-evaluated.

This function creates a knitr chunk hook that signals when one chunk's code should be saved for re-use later. The code ends up in a temporary SAS `autoexec.sas` file.

Value

There are no return values, chunk hook creation is a side effect here.

Note

If there is already an '`autoexec.sas`' in the directory where the source document is located, collected code will be added to it, and the original file will be restored after your document is processed.

Author(s)

Doug Hemken

See Also

SASmarkdown-package

Examples

````
sas_collectcode()
## Not run:
indoc <- '
---
title: "Linking SASmarkdown Code Chunks"
author: "Doug Hemken"
output: html_document
---
# An R console example
## In a first code chunk, set up with
```{r}
library(SASmarkdown)
```
## Then mark SAS code chunks with
```{sas, collectcode=TRUE}
data class;
  set sashelp.class;
  bmi = 703*weight/height**2;
  run;
```

## A later chunk that depends on the first.
````

```
```{sas}
proc means;
  var bmi;
  run;
```
'
if (!is.null(SASmarkdown::find_sas())) {
  # To run this example, remove tempdir().
  fmd <- file.path(tempdir(), "test.md")
  fhtml <- file.path(tempdir(), "test.html")

  knitr::knit(text=indoc, output=fmd)
  rmarkdown::render(fmd, "html_document", fhtml)
}

End(Not run)
```

`sas_enginesetup` *Create SAS engines for knitr*

Description

In addition to knitr's built in SAS engine, this function creates additional engines for SAS. Once created, these engines may be invoked like any other knitr engine to generate different forms of SAS output.

Set up once per session (i.e. document). Ordinarily this is run automatically when `SASmarkdown` is loaded.

Usage

```
sas_enginesetup(...)

saslog(options)
sashtml(options)
saspdf(options)
```

Arguments

`...`	arguments to be passed to `knit_engines$set(...)`. These take the form `enginename=enginefunction`
`options`	`options` are passed knitr to the engine functions when they are actually invoked within `knitr`.

Details

This is a convenience function that uses `knit_engines$set()` to define knitr language engines.

`sas_enginesetup(...)` passes it's arguments to `knit_engines$set()` in the form of `enginename=enginefunction` pairs. Three pre-defined engine functions are in this package: `sashtml`, `saslog`, and `saspdf`. These functions are used as follows.

- `sas_enginesetup(sas=saslog)` creates a language engine that returns SAS code, as well as listing output. The engine created is called "sas", and replaces `knitr`'s "sas" engine. This new engine provides better SAS error handling if you set the chunk option `error=TRUE`.
- `sas_enginesetup(saslog=saslog)` creates a language engine that returns SAS log output instead of the plain code that is usually echoed, as well as listing output. The engine created is called "saslog".
- `sas_enginesetup(sashtml=sashtml)` creates a language engine that returns SAS html output using SAS's ODS system. The engine created is called "sashtml". An additional side effect is that the html results are used "asis" - you can hide them or you can use them as ordinary document text.
- `sas_enginesetup(sashtmllog=sashtml)` creates a language engine that returns SAS log output instead of the plain code that is usually echoed, as well as html output. The engine created is called "sashtmllog".
- `sas_enginesetup(sashtml5=sashtml, sashtml5log=sashtml)` create language engines that produce html output with inline images and UTF-8 output
- `sas_enginesetup(saspdf=saspdf, saspdflog=saspdf)` create language engines that produce LaTeX output, with inline images

The end user should not need to use the language engine functions directly. These are the workhorse functions that actually call SAS and return output. Their main use is when set up within `sas_enginesetup(sashtml=sashtml`

Value

There are no return values for `sas_enginesetup`, engine creation is a side effect here.

The individual language engine functions return SAS code and SAS output internally to `knitr`.

Author(s)

Doug Hemken

See Also

`knit_engines`

Examples

```
sas_enginesetup(sashtml=sashtml, saslog=saslog)

indoc <- '
---
title: "Basic SASmarkdown Doc"
author: "Doug Hemken"
output: html_document
---
# In a first code chunk, set up with
```{r}
library(SASmarkdown)
```
Then set up SAS code chunks with
```

```
```{sas}
proc means data=sashelp.class;
run;
```
'
if (!is.null(SASmarkdown::find_sas())) {
  # To run this example, remove tempdir().
  fmd <- file.path(tempdir(), "test.md")
  fhtml <- file.path(tempdir(), "test.html")

  knitr::knit(text=indoc, output=fmd)
  rmarkdown::render(fmd, "html_document", fhtml)
}
```

---

sas_output *A function to provide cleaner output for knitr's SAS engines.*

Description

When knitr calls SAS to produce various forms of output, that output is often more cluttered than what you want to show in your SAS markdown documentation.

This function filters the output returned by SAS prior to invoking knitr's `engine_output()` function.

Usage

```
sas_output(options, code, out, extra = NULL)
```

Arguments

`options`	options object passed from a SAS engine.
`code`	code object passed from a SAS engine.
`out`	out object passed from a SAS engine.
`extra`	anything else to add to the document.

.

Details

This redefinition adds a filter to the standard `engine_output()`. At present, the same filtering is used for both SAS ODS and SAS listing output. In the future more choice and nuance could be added here. The user should not need to invoke this directly.

Value

This returns a call to `knitr::engine_output`.

Author(s)

Doug Hemken

See Also

SASmarkdown,

Examples

```
## Not run:
indoc <- '
---
title: "Basic SASmarkdown Doc"
author: "Doug Hemken"
output: html_document
---
# In a first code chunk, set up with
```{r}
library(SASmarkdown)
```
Then set up SAS code chunks with
```{sas}
proc means data=sashelp.class;
run;
```
'
if (!is.null(SASmarkdown::find_sas())) {
  # To run this example, remove tempdir().
  fmd <- file.path(tempdir(), "test.md")
  fhtml <- file.path(tempdir(), "test.html")

  knitr::knit(text=indoc, output=fmd)
  rmarkdown::render(fmd, "html_document", fhtml)
}

End(Not run)
```

---

spinsas *Convert a specially marked up SAS file to Markdown and HTML.*

Description

This function takes a SAS file containing special mark up in it's comments, and converts it to Markdown and HTML documents (or one of several other formats).

Usage

```
spinsas(sasfile, text=NULL, keep=FALSE, ...)
```

Arguments

sasfile	A character string with the name of a SAS command file, containing markup in it's comments.
text	A character string in place of a file.
keep	Whether to save intermediate files.
...	options passed to knitr::spin

Details

This function takes a SAS file containing special mark up in it's comments, and converts it into knitr's "spin" format. This is in turn sent to knitr::spin, and converted to Markdown and HTML (or one of several other formats).

Special Markup:

- "** " - Begin document text
- "*+ " - Begin chunk header
- "*R " - Begin a chunk of R code
- "*/* " - Dropped from document, ends with "*/*"

Each document element - document text, chunk headers, R code chunks, and SAS code chunks - ends with a semicolon at the end of a line.

Value

The path to the output file.

If given text instead of a file, returns the compiled document as a character string.

Author(s)

Doug Hemken

See Also

SASmarkdown-package

Examples

```
## Not run:
indoc <- '
** # Native SASmarkdown
First, call the `SASmarkdown`` package in R.;

*R require(SASmarkdown);

** Then execute some SAS code.  First specify the
chunk header, then the code.;

*+ example;
```

```
proc means data=sashelp.class;
run;
'
x<-spinsas(text=indoc)
writeLines(x, "indoc.html")

## End(Not run)
```

Index